IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of
**Bomers**

Serial No.: **10/699,968**

Filed: **November 3, 2003**

For: **Universal Computer Input Event Translator**

Attorney's Docket No: **5168-001**

Patent Pending

Examiner: Mr. Sherrod Keaton

Group Art Unit: 2175

Confirmation No.: 5026

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]**

I hereby certify that this correspondence is being:

☐ deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

☐ transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) 273-8300.

_____ _____
Date

This correspondence is being:

☒ electronically submitted via EFS-Web

# APPEAL BRIEF

## I. REAL PARTY IN INTEREST

The real party in interest is the named inventor, Florian U. Bomers.

## II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences to the best of Applicant's knowledge.

## III. STATUS OF CLAIMS

Claims 1-4 and 6-24 are pending; claim 5 is canceled. All pending claims are rejected, and all such rejections are appealed.

## IV. STATUS OF AMENDMENTS

All amendments have been entered.

## V. SUMMARY OF CLAIMED SUBJECT MATTER

The claims on appeal include independent claims 1, 11, 12, and 14. All claims relate to "input event" translation for a personal computer (PC). The present invention enables a user to select one or more types of input events of interest, associate each of those event types with one or more software-based event translators, and configure the translation behavior of each translator such that incoming input events of the selected type(s) are intercepted and processed by the associated translator(s) according to the defined translation rules. Input events are, for example, mouse clicks and movements, key presses, Musical Instrument Digital Interface (MIDI) events, USB events, and the like. As a non-limiting example, a user may configure the IDT program 10 to intercept mouse input events and re-map (translate) those events into defined keystrokes such that keyboard input may be affected by mouse movement.

***Independent Claim 1***

Claim 1 is directed to a computer readable medium storing a computer program configured for execution by a personal computer (PC). Fig. 1 depicts a PC 8 that includes system memory storing an Input Device Translator (IDT) program 10, which represents the claimed computer program.

Claim 1 includes the limitation of program instructions to create event translators that translate incoming input events to said PC into translated input events according to user-defined translation behaviors. Fig. 1 depicts external input devices 12-1,..., 12-N generating input events incoming to the PC 8 for possible event translation by the program 10, and Fig. 2 depicts an event translator 20, with paragraphs [0018] and [0019] providing supporting event translator details. Paragraph [0028] (lines 7-10) and paragraph [0029] (lines 1-6) provide example details for defining event translators 20, via drag-n-drop operations.

Claim 1 includes the further limitation of program instructions to associate each event translator with a type of incoming input event responsive to user input. Again, lines 7-10 in paragraph [0028] explain making such associations in the context of Fig. 4, which depicts a program embodiment where a user makes event translator associations via drag-n-drop operations. The Fig. 4 referred to here is the revised copy, submitted by Applicant on 6 April 2007, in response to drawing objections. Lines 1-6 in paragraph [0029] provide further details for making associations between particular input events and desired translators/translation behaviors.

Claim 1 includes the further limitation of program instructions to configure a translation behavior for each event translator responsive to user input, such that during execution of the computer program by the PC the event translator generates a desired translated input event responsive to receiving an incoming input event to said PC of the type of incoming input event associated with the event translator, including program instructions to define a translation function that modifies incoming input events to said PC according to one or more user-configured functions. Paragraphs [0028] and [0029] explain one example of user-based configuration of event translation behaviors, done with reference to Fig. 4 (revised sheet from 6 April 2007) wherein a user "drops" pre-defined or custom translation rules into a given event translator 20. Paragraph [0030], lines 1+, provide further event translator configuration details, and paragraph [0036] provides a series of specific configuration examples.

***Independent Claim 11***

Claim 11 is directed to a method of adapting a personal computer (PC) such that its response to one or more types of input events is modified according to user-configured event translation behavior. The claim includes the limitation of defining one or more event translators for execution by said PC, wherein each event translator maps incoming input events to said PC of a selected type into translated input events according to a defined translation behavior, and

wherein the defined translation behavior includes modifying one or more event parameters of the incoming input events. Paragraphs [0028] and [0029] give event translator configuration examples in the context of Fig. 4, wherein the IDT program 10 connects a type of input event to a given event translator 20 in response to user input, to configure the type of input events to be translated by the given event translator 20.

Claim 11 includes the further limitation of configuring the defined translation behavior for each event translator based on user input to the PC. As explained in paragraphs [0028] and [0029], the IDT program 10 in one or more embodiments provides a graphical user interface 120, which enables a user to drag-n-drop custom or pre-defined rules into a given event translator 20, to define its translation behaviors. More particularly, from Fig. 4, one sees that input events field 121 and input events selector field 126 allows a user to define or identify input events to be processed by type (e.g., mouse, keyboard, etc.), and then graphically connect (or "map") particular types of input events to particular event translators 20 within a translation rules field 122. See paragraph [0028] for supporting explanations. In turn, as explained in paragraph [0029], a user drags desired event translation rules from rules field 128, and drops them into particular ones of the event translators 20, to define the behaviors of those translators.

Finally, claim 11 includes the limitation of detecting input events of the selected types incoming to the PC and translating those incoming input events into corresponding translated input events according to the defined translation behaviors of the one or more event translators. This process is illustrated in Fig. 3, which depicts processing incoming input events from a mouse, keyboard, etc., to determine whether they match types of events for which event translators 20 have defined. See lines 1-2 of paragraph [0024], explaining that the IDT program 10 detects input events based on monitoring or otherwise intercepting them, and see the whole of paragraph [0025] for its corresponding example discussion of event translation.

***Independent Claim 12***

Claim 12 is directed to a method of modifying input event behavior in a personal computer (PC), e.g., the PC 8 of Fig. 1. According to claim 12, the method includes defining one or more event translators for execution by the PC and associating each event translator with a selected type of incoming input event to said PC, responsive to input by a user. Fig. 4 and paragraphs [0028]-[0030] provide example details for a user to define event translators 20 by dropping event types from field 126 into field 120, and to map those event types to particular event translators 20 in field 122, based on graphically connecting them.

Claim 12 also claims defining a translation behavior of each event translator responsive to input by a user, including defining a translation function that modifies incoming input events to the PC according to one or more user-configured functions. Lines 6-9 of paragraph [0029] explain that a user defines an event translation flow for a given event translator 20 by dragging-and-dropping rules from rules field 128 into (graphically) depicted event translators 20. Further, the whole of paragraph [0030] details how a user sets general actions for each depicted event translator 20 via "general actions" editor field 130 in Fig. 4.

Finally, claim 12 includes the limitation of generating translated input events in said PC based on executing associated ones of the event translators responsive to detecting incoming input events to said PC of the selected types. Fig. 3 depicts the generation of translated input events (at block 104), which is discussed at lines 1-4 of paragraph [0025]. The generation of translated input events is also depicted in Figs. 1 and 2, and described in paragraphs [0020] and [0021].

***Independent Claim 14***

Claim 14 is directed to a computer readable medium storing a computer program, the computer program configured for execution on a personal computer (PC). Fig. 1 depicts a PC 8 that includes system memory storing an Input Device Translator (IDT) program 10, which represents the claimed computer program.

Claim 14 includes the limitation of program instructions to enable a user to select a type of input event to the PC from a plurality of input event types. Input events selector field 126 in Fig. 4 depicts a selectable list or set of input event types, and an example of user-based selection is described in lines 5-10 of paragraph [0028].

Claim 14 also includes the limitation of program instructions to determine whether a given input event to the PC occurring during execution of the computer program by the PC matches the selected type of input event, and program instructions to perform a desired input event translation. The claimed translation is based on processing the given input according to one or more input event translation rules if the given input event matches the selected type of input event. The desired input event translation includes a translation function that modifies incoming input events according to one or more user-configured functions. Fig. 3 illustrates the claimed type matching and subsequent event translation at steps 102 and 104. The beginning of paragraph [0025] describes these steps.

As for the user-configured functions, Fig. 4 depicts a rules processor field 128 that allows a user to define one or more functions to be applied to incoming input events by a given event translator 20. Fig. 4 explicitly shows an example function of the form $f(x) = t_x$ +/- delta t. Further, lines 6-11 of paragraph [0017] state that a user can configure the IDT program 10 to translate input events according to a mathematical function, such as to invert or smooth mouse movement input events, or to time-delay them. (The function shown in Fig. 4 is a time delay function.) Lines 8 and 9 in paragraph [0020] also call out example mathematical functions that can be applied for smoothing, amplifying, attenuating, etc.

## VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-4, 6-8, 11-14, and 16-22 are rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Pub. 2004/0263477 A1 (hereinafter "Davenport"), in view of U.S. Pub. 2004/0257341 A1 (hereinafter "Bear").

Claims 9 and 10 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Davenport and Bear, in further view of U.S. Pub. 2003/0071842 A1 (hereinafter "King").

Claim 23 is rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement.

Claim 23 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

## VII. ARGUMENT

### A. Claim 23 is enabled

To satisfy the enablement requirement of 35 U.S.C. § 112, first paragraph, "the specification of a patent must teach those skilled in the art how to make and use the full scope of the claimed invention without 'undue experimentation'". *In re Wright*, 999 F.2d 1557, 1561, 27 U.S.P.Q.2d 1510 (Fed. Cir. 1993). Notably, "[s]ome experimentation, even a considerable amount, is not 'undue' if, e.g., it is merely routine, or if the specification provides a reasonable amount of guidance as to the direction in which the experimentation should proceed. *In re Wands*, 858 F.2d 731, 737, 8 U.S.P.Q.2d 1400 (Fed. Cir. 1988).

Regarding the alleged non-enablement of claim 23, in Item 2 of the Final Office Action ("FOA"), the examiner states that:

> The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. The applicant has failed to disclose what the user configured mathematical scaling is comprised of.

The applicants have disclosed a mathematical function but do not disclose if mathematical scaling is configured within this function.

Claim 23 depends from claim 21, which depends from claim 1, and it stipulates that "the program instructions to modify one or more event parameters of the incoming input events comprise program instructions to apply a user-configured mathematical scaling to one or more event parameters of a user-selected type of incoming input event, to thereby create corresponding translated input events of the same user-selected type, but with one or more scaled event parameters."

As a first point, paragraph [0010] of the filed application stipulates that an embodiment of input event translation includes scaling. More generally, the application of a mathematical scaling to one or more event parameters of an incoming event is supported in the filed application at least at paragraphs [0010], [0017], and [0020] (attenuating). As the Board of Patent Appeals & Interferences recently reiterated, "the specification need only teach those aspects of the invention that one skilled in the art could not figure out without undue experimentation." *Ex Parte Mario Boschetti and Claudio Boschetti*, Appeal No. 2008-1397, 2008 WL 4418250 (Bd.Pat.App. & Interf.).

As a second point, the term "scaling" is well understood by those skilled in the art. For example, Wikipedia defines scaling as a linear transformation that enlarges or diminishes objects. (See http://en.wikipedia.org/wiki/Scaling.) That meaning is consistent with "scaling" as used in the filed application, where attenuation is given as an example of the mathematical functions that can be applied to an input event. (See paragraph [0010].) Applicant submits that the evidence demonstrates that the full scope of the term "mathematical scaling" as used in the claim is enabled by the specification as filed, and that those skilled in the art would have no trouble practicing the claimed invention without undue experimentation.

The scaling limitation of claim 23 is clearly enabled and no undue experimentation would be required to practice it.

**B. Claim 23 is definite**

According to a recent decision by the Board of Patent Appeals and Interferences, the Patent Office employs "...a lower threshold of ambiguity when reviewing a pending claim for indefiniteness than those used by post-issuance reviewing courts." Particularly, the Patent Office is justified in holding a claim indefinite if it "...is amenable to two or more plausible claim constructions." *Ex parte Kenichi Miyazaki*, 2008 WL 5105055 (Bd.Pat.App. & Interf.) This lowered threshold contrasts with, e.g., the indefinite-only-if-insolubly-ambiguous standard articulated by the Federal Circuit in *Datamize, LLC v. Plumtree Software, Inc.*, 417 F.3d 1342, 1347, 75 U.S.P.Q.2d 1801 (Fed. Cir. 2005).

The examiner argues that what is meant by "mathematical scaling" is unclear and, hence, indefinite. However, as a general proposition, mathematical scaling is well understood by those skilled in the art, and the filed specification underscores that meaning by highlighting attenuation (which is understood to be a type of fractional scaling) as an example. For these reasons, Applicant believes that claim 23 is definite.

**C. Independent claims 1, 11, 12, and 14, and their dependent claims 2-4, 6-8, 13, and 16-22 are not obvious**

The examiner rejects independent claims 1, 11, 12, and 14 as obvious over the combination of Davenport and Bear. Davenport allegedly teaches all computer input event translation limitations of claims 1, 11, 12, and 14, except for implementing such translations in a personal computer (PC). Bear is not alleged as teaching any specific aspect of these independent claims, and instead appears to be used merely as an example of a PC-based

program. On that basis, the examiner argues that it would have been obvious in a legal sense to combine Bear with Davenport, to obtain the invention of claims 1, 11, 12, and 14.

Against claims 1, 11, 12, and 14, the examiner states that one skilled in the art would have been motivated (from Bear's teachings) to implement Davenport in a PC "because it improves the efficiency of the system being that the PC provides more powerful processors." In contrast Davenport teaches that a programmed microcontroller 21 of its external input device 20 directly implements Davenport's disclosed functionality. (See Fig. 4a, for example.) One does not need a PC processor (more powerful, or otherwise) to implement Davenport, and no one skilled in the art would understand that a PC is needed.

However, a key legal failing of the obviousness rejection is that Davenport explicitly teaches that its inventive value derives from the implementation of event processing external to any computer or game console. Paragraph 0061 of Davenport states that "[i]t is a significant and salient aspect of the invention that the programmed actions and sequences are stored in non-volatile memory associated with the microcontroller 21, and thus are incorporated into the invention *independently of any computer or game to which the invention 20 may be connected.*" When one considers that Bear relates to a PC-based program that provides an enhanced user navigation interface, and is unrelated to the input event processing at issue in Davenport, and that Davenport explicitly specifies implementation outside of and independent from a PC or other computer, it is clear that no motivation to combine Davenport and Bear exists.

In more detail, Davenport at paragraph 0029 states that it is significant that the microcontroller and its programming and tables (i.e., the peripheral action language) of Davenport's invention all are allocated within the external input device 20, and that "...the actions generated by the invention are independent of the computer to which it is connected." It therefore is legal error for the examiner to state that one skilled would find it obvious from the teachings in Bear to implement Davenport as a PC program configured for execution in a PC.

Davenport repeatedly teaches that its key advantages are obtained by implementing an input device 20 independent from any computer to which it is attached. Moreover, as noted, Bear is not directed to translating peripheral input events, but rather is directed to improving computer "navigation" in comparison to basic mouse/keyboard approaches. (See Abstract, Summary.) Bear is used merely for its explicit statement that Bear's invention is implemented within a PC (e.g., see paragraphs 0074, 130, and 131, as referenced by the examiner).

In further error, the examiner asserts that "Davenport has also disclosed that the peripheral action language (PAL) can be embodied outside of the input device (Figure 4b; Page 4, Paragraph 0058)." This statement apparently argues that Davenport itself suggests implementing its peripheral action processing somewhere other than the disclosed input device 20. However, paragraph 0058 and Fig. 4b of Davenport explicitly illustrate a standalone embodiment of the input device 20, which is separate from a computer—see the "TO/FROM" labeling at the rightmost side of Fig. 4b and paragraph 0045. Rather than demonstrating that Davenport could be implemented within a PC, Fig. 4b reinforces the clear teachings in Davenport that its input event processing is independent of and external to a PC.

In view of the above analysis, Applicant submits that the rejection of claims 1, 11, 12, and 14 under section 103(a) is erroneous because the teachings of Davenport and Bear would not have motivated one of ordinary skill in the art to modify Davenport's external input device 20 in a manner which would meet the limitations of claim 1, 11, 12, or 14. As reinforced by the Supreme Court of the United States, "rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness". *KSR Intern. Co. v. Teleflex Inc.*, 550 U.S. 398, 127 S.Ct. 1727, 1740 (quoting *In re Kahn*, 441 F.3d 977, 988, 78 U.S.P.Q.2d 1329 (Fed. Cir. 2006)).

Here, it appears that the articulated reasoning amounts to the examiner finding in Bear various statements regarding its implementation as a PC program, and using those statements

as evidence that it would have been obvious to modify Davenport to obtain the invention of claims 1, 11, 12, and 14. Bear, however, does not relate to Davenport in any meaningful way, and Davenport explicitly defines its invention as being separate from, external to, and independent of any computer or game system.

These differences are important because obviousness is resolved according to underlying factual determinations, which include determining "(1) the scope and content of the prior art, (2) any differences between the claimed subject matter and the prior art, (3) the level of ordinary skill in the art, and (4) where in evidence, so-called secondary considerations." *Graham v. John Deere Co.*, 383 U.S. 1, 17-18, 148 U.S.P.Q. 459 (1966). The examiner appears not to recognize that Bear, which relates to an improved user navigation interface, has no real relevance to Davenport, which relates to a stand-alone input device for preprocessing computer or game console inputs. As such, the examiner overlooks the fact that the teachings in Bear would not, in the view of one of ordinary skill in the art, combine with Davenport to produce the claimed invention. Applicant submits therefore that the argued-for combination of Bear and Davenport do not render claims 1, 11, 12, and 14 legally obvious. Further, as claims 2-4, 6-8, 13, and 16-22 depend from corresponding ones of these independent claims, none of these dependent claims, which add further narrowing limitations, are obvious over Davenport and Bear.

**D. Dependent claims 2 and 3 are not obvious**

Applicant believes that the rejection of claims 2 and 3 as obvious over the combination of Davenport and Bear rests on plain factual error. These claims stipulate that the input event translation program instructions of claim 1 include instructions to create (PC) operating system hooks to detect event messages associated with incoming input events. The claim-2 rejection argument on pp. 4 and 5 of the FOA vaguely states that Bear "provides the operating system," but ignores the fact that Davenport is external to a PC and includes no discussion of hooks.

Because Davenport explicitly teaches a separate, external input device 20, which includes a programmed microcontroller 21, Bear cannot be understood as "providing" an operating system to Davenport. Further, the examiner's rejection arguments fail to note that Bear actually discusses various types of "hooks" at paragraphs 0007, 0116, 0117, and 0118. However, none of these "hook" related teachings in Bear teach or suggest the limitations of claims 2 and 3; nor do such teachings appear to relate to Davenport in any understandable manner.

In more detail, claim 2 claims that input event translation instructions of claim 1 include instructions to create operating system hooks, to detect input event messages. Claim 3 is similar, but claims instructions to create operating system hooks to detect input events. Bear does not teach or suggest operating system hooks to detect input events or input event messages, nor does Davenport teach or suggest such limitations. Indeed, Davenport can fairly be argued as teaching away from any operating system related details in Bear, as Davenport explicitly calls out its implementation separate and independent from a computer (and, therefore, independent of the computer's operating system).

**E. Dependent claims 6, 17 and 24 are not obvious**

Claims 6 and 17 include explicit limitations to program instructions directed to "swallowing" an incoming input event. The filed application (paragraph 0022) explains that a "swallowed" event is hidden from other programs/processes running on the computer. The examiner has failed to make a *prima facie* case for the obviousness of claims 6 and 17, because neither Davenport nor Bear disclose the claimed swallowing. The examiner erroneously suggests that Davenport's event linking at paragraph 0066 is the claimed swallowing. Paragraph 0066 says nothing about swallowing an input event and instead discusses processing (mouse) button clicks (one click, two clicks, one click generating two

actions, etc.). It is clear factual error to argue on pp. 6 and 12 of the FOA that linking actions in Davenport means that "one of the actions is swallowed."

Similarly, on p. 12 of the FOA, the examiner argues with respect to claim 17 that linking events in Davenport means that one event is swallowed. This argument represents a legally impermissible construction of the claim term "swallow." As the filed specification teaches, swallowing an input event means hiding it from other programs/processes running on the PC (e.g., hiding it as if it did not happen). In paragraph [0066], Davenport discusses a form of linking input events, but such linking is done to relate actions or to generate multiple actions, not to swallow (hide) an input event.

Further, as regarding claim 24, the examiner on p. 14 in the FOA uses yet another argument as to how Davenport discloses swallowing. On p. 14, the examiner states that "Davenport discloses the ability of a check in paragraph 71, therefore a swallow functionality is provided." Applicant cannot discern what the examiner means by saying paragraph [0071] of Davenport discloses a check, as that paragraph apparently relates to improving 8-way scrolling and is silent with respect to input event swallowing.

The examiner also states on p. 14 that it "would have been obvious to swallow the input event any [sic] number of reasons that are needed for the program in use." This statement is not supported by the record; Davenport does not disclose the claimed swallowing, nor does Bear, nor has the examiner provided any evidence that event swallowing in the manner claimed is known or otherwise obvious as a general proposition. This misunderstanding of what is needed to make a legal showing of obviousness also appears in the comments on p. 16 of the FOA. There, the examiner states that "[p]er claim 6, Applicants argue the meaning of swallowing, examiner disagrees. The swallowing event can take on many forms, plus applicant has only proved exemplary definitions in the spec and the claims recite none of the detailed limitations argued or disclosed."

"Swallowing" as used in the claims is defined in the filed application as hiding an input event from other programs/processes running on a PC. It is unclear, therefore, what the examiner means by saying that the "swallowing event can take on many forms" or that "applicant has proved only exemplary definitions in the spec." Applicant claimed event swallowing in claims 6, 17, and 24, and defined what was meant by swallowing in the filed application. Neither Bear nor Davenport teach or suggest the claimed event swallowing and Applicant submits that these claims are not obvious.

**F. Dependent claim 8 is not obvious**

Claim 8 depends from independent claim 1, and includes limitations directed to determining whether an incoming input event triggers an activation of or a focus shift to a targeted program. The examiner rejects claim 8 as obvious over Davenport and Bear, but neither reference includes any discussion relevant to the claimed limitation. Bear does discuss focus (in a different context), but the examiner's rejection refers only to paragraphs 0070-0072 in Davenport for rejection support. Those paragraphs discuss 8-way scroll wheel controls and say nothing remotely relevant to the claim limitation. The multiple instances in Bear where focus is discussed (ignored by the examiner) apparently relate to navigation element activation rather than to the claim limitations.

Regarding Applicant's rebuttal arguments for claim 8, the examiner responds on p. 16 of the FOA by stating that:

> Examiner disagrees. Davenport shows matches [sic] of actions in paragraph 72, therefore the matches of the action within software can provide activations of targeted programs within the body of the software.

The precise meaning of the examiner's statement is unclear. What is clear is that claim 8 depends from claim 1 and stipulates "wherein program instructions to enable a user to configure a translation behavior for each event translator responsive to user input comprises program instructions to determine whether the incoming input event triggers an activation of or a focus

shift to a targeted program." (Emphasis added.) Neither Davenport nor Bear teache program instructions enabling a user to configure event translation behavior for an event translator that allows the event translator to determine whether the incoming input event triggers an activation of or a focus shift to a targeted program.

**G. Dependent claim 13 is not obvious**

Claim 13 includes express limitations to detecting operating system events of a PC that are associated with selected types of incoming input events, and translating those input events according to defined translation behaviors. The rejection argument on p. 9 of the FOA states that Bear inherently provides a computer operating system and states that Davenport teaches everything else.

The argument ignores the fact that Davenport is expressly taught as being implemented external to and independent from any computer or game system and that Bear, therefore, cannot be argued as providing a PC operating system to Davenport. Further, the rejection overlooks the express limitations in the claim, namely, that "executing associated ones of the event translators responsive to detecting incoming input events to said PC of the selected types comprises: detecting operating system events of said PC that are associated with the selected types of incoming input events; and for each detected incoming input event of a selected type, translating that incoming input event according to the translation behavior defined for the associated event translator or translators."

In his specific rejection of claim 13 on p. 9 of the FOA, the examiner states that:

> ...detecting operating system events to said PC(Bear: Inherently provides an OS within its computer system Page 3, Paragraph 74-75) that are associated with the selected types of incoming input events; and for each detected incoming input event of a selected type, translating that incoming input event according to the translation behavior defined for the associated event translator or translators(Davenport: Page 4, Paragraph 59 and 64; Page 5, Paragraph 67).

In addition to reiterating the point that Bear's operating system does not "combine" or integrate with Davenport's input device 20, the above argument fails because neither Davenport nor Bear teach or suggest the claimed detection of operating system events and translation of the input events that are associated with them.

**H. Dependent claim 15 is not obvious**

Claim 15 depends from independent claim 14 and stipulates that the computer program of claim 14 is a WINDOWS-based program. The rejection of claim 15 is based on "Official Notice" that WINDOWS programs are notoriously well known. Applicant previously complained that Office Notice was inappropriate. In response, on p. 11 of the FOA, the examiner states that:

> However Davenport and Bear do utilize a PC to interact with the invention and official notice is taken that windows and windows based programs are notoriously well known operating systems for PC's.

And on p. 17, the examiner further states that:

> Examiners [sic] official notice is taken to show how windows and windows based programs are well known in the art. Examiner also notes that the davenport [sic] may stress the portability but also discloses the ability to provide a system can be separate of the input device and therefore that system can be easily incorporated within in a pc as an add-on within that windows functionality.

These response arguments by the examiner do not address the primary complaint regarding the use of Official Notice. That is, the issue is not whether WINDOWS programs are notoriously well known, but whether the argued-for combination of Davenport and Bear teach the limitations of claim 15, which depends from independent claim 14.

In this regard, the examiner's statements are demonstrably wrong. For example, Davenport does not disclose that its "system" can be separate of Davenport's "input device." Davenport's invention is its input device 20. Figs. 4a and 4b in Davenport both explicitly illustrate that the input device 20, including the microcontroller 21 and the PAL (event language) are implemented outside and apart from a computer. Further, paragraph [0025] of Davenport states that the input device of Davenport can be standalone or can be integrated into a

peripheral controller, which by definition is external to a PC. Still further, paragraphs [0029] and [0061] of Davenport both expressly state that it is a significant and salient aspect of Davenport's invention (system in the examiner's parlance) that the input device and its PAL (event language) are independent of any computer or game to which the input device is attached.

Davenport does not teach or suggest a WINDOWS-based event translation program for execution within a PC, as claimed in claim 15, nor does Bear. Further, Davenport expressly teaches against implementing its "system" within a computer and rejecting claim 15 on the basis that, in general, WINDOWS programs are notoriously well known is legal error.

**I. Dependent claim 20 is not obvious**

Claim 20 explicitly claims that the computer program of independent claim 14 includes program instructions to time-delay input events of selected types according to a desired time delay value. The rejection argument on p. 13 of the FOA states that paragraph 0066 in Davenport teaches this limitation. That argument represents clear error. Paragraph 0066 in Davenport states that one mouse click can generate two actions (one action on the button press, and another action on the button release). The paragraph also states that a mouse user (i.e., a human operator) can control the time delay between press and release by simply holding down the mouse button for a desired amount of time. Davenport's description of a human user delaying mouse button releases is a legally improper basis for rejecting claim 20.

In more detail, claim 20 depends from independent claim 14 and stipulates that the program instructions to perform a desired input event translation comprise program instructions to time-delay input events of the selected type according to a desired time delay value. One sees a functional example of this in the rules processing selector field 128 of Fig. 4, and paragraph [0017] (among others) explains that the IDT program 10 of the present invention includes instructions allowing a user to configure the time delay imparted to input events, as part of defining input event translation behavior.

Against this claimed automated, computer-program implemented event translation delay, the examiner states on pp. 12 and 13 of the FOA that Davenport teaches:

> ...discloses wherein the program instructions to perform a desired input event translation by processing the given input according to one or more input event translation rules if the given input event matches the selected type of input event comprise program instructions to time-delay input events of the selected type according to a desired time delay value (Davenport: Page 5, Paragraph 66). Allows [sic] user to change time interval.

(Emphasis added.) The emphasized portion underscores the error in the rejection logic. Davenport does not provide an event translation program within the meaning of claim 20, including program instructions enabling a user to configure a desired translation delay, but rather simply describes that a user can pause for a desired length of time between pressing a mouse button and releasing it, and can therefore manually control the time between button click and button release events.

## VIII. CLAIMS APPENDIX

The following claims are on appeal:

1. A computer readable medium storing a computer program configured for execution by a personal computer (PC) and comprising:

program instructions to create event translators that translate incoming input events to said PC into translated input events according to user-defined translation behaviors;

program instructions to associate each event translator with a type of incoming input event responsive to user input; and

program instructions to configure a translation behavior for each event translator responsive to user input, such that during execution of the computer program by the PC the event translator generates a desired translated input event responsive to receiving an incoming input event to said PC of the type of incoming input event associated with the event translator, including program instructions to define a translation function that modifies incoming input events to said PC according to one or more user-configured functions.

2. The computer readable medium storing a computer program of claim 1, wherein program instructions to create event translators that translate incoming input events to said PC into translated input events according to user-defined translation behaviors comprise program instructions to create one or more operating system hooks to detect event messages associated with incoming input events corresponding to one or more types of computer input devices.

3. The computer readable medium storing a computer program of claim 1, wherein program instructions to create event translators that translate incoming input events to said PC into translated input events according to user-defined translation behaviors comprise program

instructions to create one or more operating system hooks to receive event messages associated with incoming input events corresponding to one or more types of computer input devices.

4. The computer readable medium storing a computer program of claim 1, wherein program instructions to configure a translation behavior for each event translator responsive to user input comprises program instructions to define an incoming-to-translated input event mapping that sets the type of translated input event to be generated.

6. The computer readable medium storing a computer program of claim 1, wherein program instructions to configure a translation behavior for each event translator responsive to user input comprises program instructions to determine whether the incoming input event is swallowed or passed-through.

7. The computer readable medium storing a computer program of claim 1, wherein program instructions to configure a translation behavior for each event translator responsive to user input comprises program instructions to determine whether the incoming input event causes a one-shot translated input event or causes a repeating translated input event.

8. The computer readable medium storing a computer program of claim 1, wherein program instructions to enable a user to configure a translation behavior for each event translator responsive to user input comprises program instructions to determine whether the incoming input event triggers an activation of or a focus shift to a targeted program.

9. The computer readable medium storing a computer program of claim 1, further comprising program instructions to display a graphical user interface on a display screen of said

PC, and wherein the graphical user interface is configured to enable a user to graphically define one or more event translators, and graphically link one or more selected incoming input events to one or more translated input events through the one or more graphically defined event translators.

10. The computer readable medium storing a computer program of claim 9, wherein the graphical user interface is configured to present a user with visual depictions of available incoming input event types and to enable the user to drag-n-drop selected ones of those incoming input event types into an input event field, and into a translated input event field, and to make desired event translation connections between respective incoming input events in the input event field and respective translated input events in the translated input event field.

11. A method of adapting a personal computer (PC) such that its response to one or more types of input events is modified according to user-configured event translation behavior, the method comprising:

defining one or more event translators for execution by said PC, wherein each event translator maps incoming input events to said PC of a selected type into translated input events according to a defined translation behavior, wherein the defined translation behavior includes modifying one or more event parameters of the incoming input events;

configuring the defined translation behavior for each event translator based on user input to the PC; and

detecting input events of the selected types incoming to the PC and translating those incoming input events into corresponding translated input events according to the defined translation behaviors of the one or more event translators.

12. A method of modifying input event behavior in a personal computer (PC), the method comprising:

defining one or more event translators for execution by said PC and associating each event translator with a selected type of incoming input event to said PC, responsive to input by a user;

defining a translation behavior of each event translator responsive to input by a user, including defining a translation function that modifies incoming input events to said PC according to one or more user-configured functions; and

generating translated input events in said PC based on executing associated ones of the event translators responsive to detecting incoming input events to said PC of the selected types.

13. The method of claim 12, wherein generating translated input events in said PC based on executing associated ones of the event translators responsive to detecting incoming input events to said PC of the selected types comprises:

detecting operating system events of said PC that are associated with the selected types of incoming input events; and

for each detected incoming input event of a selected type, translating that incoming input event according to the translation behavior defined for the associated event translator or translators.

14. A computer readable medium storing a computer program, the computer program configured for execution on a personal computer (PC) and comprising:

program instructions to enable a user to select a type of input event to said PC from a plurality of input event types;

program instructions to determine whether a given input event to said PC occurring during execution of the computer program by said PC matches the selected type of input event; and

program instructions to perform a desired input event translation by processing the given input according to one or more input event translation rules if the given input event matches the selected type of input event, said desired input event translation including a translation function that modifies incoming input events according to one or more user-configured functions.

15. The computer readable medium storing a computer program of claim 14, wherein the computer program comprises a WINDOWS-based program configured for execution on a WINDOWS-based PC.

16. The computer readable medium storing a computer program of claim 14, wherein the program instructions to enable a user to select a type of input event to said PC from a plurality of input event types comprise program instructions to enable selection from a plurality of event types include two or more of mouse events, keyboard events, MIDI events, Universal Serial Bus device events, RS-232 serial bus events, game port events, audio input events, analog input events, and infrared port events.

17. The computer readable medium storing a computer program of claim 14, wherein the program instructions program instructions to perform a desired input event translation by processing the given input according to one or more input event translation rules if the given input event matches the selected type of input event comprise program instructions to perform one or more of a plurality of translations comprising a re-mapping of the given input event type

to one or more other input event types, a time-delay of the given input event, a parameter modification of the given input event, a swallowing of the given input event to hide it from one or more other computer processes, and a swallowing of the given input event to hide it from additional event translation processing.

18. The computer readable medium storing a computer program of claim 14, wherein the program instructions to perform a desired input event translation by processing the given input according to one or more input event translation rules if the given input event matches the selected type of input event comprise program instructions to re-map input events of the selected type into input events of at least one other type.

19. The computer readable medium storing a computer program of claim 14, wherein said translation function that modifies incoming input events according to one or more user-configured functions comprises program instructions to modify one or more event parameters of input events of the selected type.

20. The computer readable medium storing a computer program of claim 14, wherein the program instructions to perform a desired input event translation by processing the given input according to one or more input event translation rules if the given input event matches the selected type of input event comprise program instructions to time-delay input events of the selected type according to a desired time delay value.

21. The computer readable medium storing a computer program of claim 1, wherein including program instructions to define a translation function that modifies incoming input events according to one or more user-configured functions comprises including program instructions to modify one or more event parameters of the incoming input events.

22. The computer readable medium storing a computer program of claim 21, wherein the program instructions to modify one or more event parameters of the incoming input events comprise program instructions to apply a user-configured mathematical function at least to selected types of incoming input events.

23. The computer readable medium storing a computer program of claim 21, wherein the program instructions to modify one or more event parameters of the incoming input events comprise program instructions to apply a user-configured mathematical scaling to one or more event parameters of a user-selected type of incoming input event, to thereby create corresponding translated input events of the same user-selected type, but with one or more scaled event parameters.

24. The computer readable medium storing a computer program of claim 14, wherein the program instructions program instructions to perform a desired input event translation by processing the given input according to one or more input event translation rules if the given input event matches the selected type of input event comprise program instructions to swallow the given input event to hide it from one or more other computer processes, or to swallow the given input event to hide it from additional event translation processing.

## IX. EVIDENCE APPENDIX

**NONE**

## X. RELATED PROCEEDINGS APPENDIX

**NONE**

Respectfully submitted,

COATS & BENNETT, P.L.L.C.

Dated: December 31, 2008

Michael D. Murphy
Registration No.: 44,958

1400 Crescent Green, Suite 300
Cary, NC 27518
Telephone: (919) 854-1844
Facsimile: (919) 854-2084